REMARKS

Claims 58-73 are pending and at issue as of the December 20, 2005 office action. Of the pending claims, all stand rejected as either unpatentable under 35 U.S.C. §102(e) based on Mathur et al. (USPN 6,671,745) or unpatentable under 35 U.S.C. §103 based on Mathur et al. in combination with one or more of Brunner et al. (USPN 4,727,544), Pascal (USPN 5,791,851), and Angelo (USPN 5,944,821). Of these claims, claims 58 and 71 are independent, and have been amended above. As discussed in the remarks to follow, applicants respectfully assert that none of the prior art teaches the now recited subject matter. By amendment above applicants have also added claims 74 and 75 which depend from claim 58 and are also distinct from the prior art by implication, and by recitation.

As an initial matter, applicants would like to express their appreciation to the examiner for the consideration of the continued examination request contemporaneously filed herewith. While applicants have disagreed with the substance of the rejections thus far, applicants nonetheless appreciate the examiner's attention to this application and ask that the examiner feel free to contact the below-signed representatives if the examiner deems that may be helpful to granting allowance in this case.

Prior Art Rejections

With respect to independent claim 58, the office action rejects this claim based on Mathur et al., which the office action deems as teaching, *inter alia*, the recited system handler application. The recited system handler application, however, is not only "operable to initiate a game based on data variables stored in the nonvolatile storage," as described in the office action. In the amended form, the system handler application is "operable to load at least one of the gaming program shared objects in response to a change in the stored game data variables by another of the at least one gaming program shared objects." In other words, not only does the system handler application have initiation features for starting a wagering game, the system handler application is also capable of loading different gaming program shared objects of that game based on changes in stored game data. In this way, for example, a system may be designed where the entire code for a wagering game need not be loaded at one time and linked for execution of various parts in response to an event. Instead, only that

code that is needed to execute at any given time may be loaded and in response to changes in the game.

The office action's rejection of claim 58 points to various portions of Mathur et al. where, among other things, components, modules and APIs are discussed. As described, modules are major functional blocks of an operating system and expose APIs to applications so that applications may interface and call the modules. Components are groups of functions that provide capabilities on a smaller scale than modules and expose their own APIs for call by applications, modules, or other components.

Setting aside the pertinence of these descriptions to the recitations of claim 58, what Mathur et al. clearly does not teach or suggest is a system handler application that is "operable to load at least one of the gaming program shared objects in response to a change in the stored game data variables by another of the at least one gaming program shared objects," as recited in amended claim 58. Indeed, the office action would appear on its face to agree with as much, as the only other similar recitation from the claims was the reference to an ability to execute a corresponding callback function based on changing a data variable in nonvolatile storage in claim 63, and the office action recognized that Mathur et al. alone did not teach such subject matter.

The examiner rejected claim 63 as obvious over Mathur et al. in combination with Pascal. The office action describes Pascal as teaching a gaming device "wherein callbacks are employed to communicate information between application modules upon the occurrence of certain events." Office Action, page 6. Pascal, of course, describes an operating system having multiple subsystems each for performing a distinct function and which register with other subsystems so that they can be properly notified (through a callback) upon the occurrence of some event. The Pascal system for example allows subsystems to complete their tasks when a fault is identified and communicated to the subsystem, without commencing further additional activities.

There is no suggestion in Pascal, however, of the claimed subject matter. There is no suggestion of the recited system handler application, which in addition to being operable to initiate a game in response to stored game data variables and to containing APIs

that may be called by gaming program shared objects of that game is able to load and execute particular gaming program shared objects in response to changes in the stored game data variables. Indeed, in reference to Figures 3A-3E, Pascal describes that there are two ways of launching an application, a normal way in which the entire application is loaded at once allowing for multiple parallel applications and an interrupting way in which any currently executing application is first paused and then a second, different application is executed.

In contrast to these teachings of Pascal, the present application allows for selective execution of different shared objects of a wagering game depending on the events occurring during that game, such as via a change in stored game data variables or an event occurring via a device handler. The entire wagering game need not be loaded at once for execution, but rather only the currently used shared object of that game need execute. That shared object is operable to interact with a system handler application for API calls and device handling, but when another aspect of that same wagering game, another operation or segment altogether, is needed, then the system handler application is capable of loading the next gaming program shared object dynamically, i.e., during runtime. In that way, system resources need not be excessively large for proper execution. And in that way, game integrity/verification may be better achieved.

In short, none of the prior art whether taken alone or in combination teaches a system handler application "operable to load at least one of the gaming program shared objects in response to a change in the stored game data variables by another of the at least one gaming program shared objects," as now recited in claim 58. For this reason alone, the rejections of claim 58 and dependent claims 59-70 are traversed.

Claims 74 and 75 have been added by amendment above and also depend from claim 58. Therefore, these claims are in condition for allowance by virtue of their dependence from claim 58. Further, claim 74 recites that the wagering game comprises "a plurality of segments each comprising a gaming program shared object" and wherein "the system handler is operable to dynamically change the wagering game from one of the plurality of segments to another of the plurality of segments in response to the change in the stored game data variables." As described by way of example on page 13, the use of a system handler responsive to stored game data variables may allow for adapting or changing

the wagering game itself by accessing different APIs depending on the wagering game. A machine may have many gaming program objects that may be available to the different games that can theoretically run on that machine. The machine may be able to implement different wagering games, each comprising a plurality of segments and gaming program shared objects, by accessing different gaming program shared objects in response to changes in stored game data variables, and doing so during runtime or in a programmed manner. In this way, the wagering game itself may change from one game operation to another, in response to changes in stored game data variables. None of the cited art teaches or suggests such subject matter. Dependent claims 74 and 75 are in condition for allowance for these separate reasons as well.

With respect to independent claim 71, the office action points to Mathur et al. as anticipating the claim. The office action points to discussions in Mathur et al. regarding the interaction between an application and a core system interface 220 and the device manager 210 and device drivers. Yet, the office action appears to consider only where the examiner believes that the system handler application recitations in the claim are taught by Mathur et al. The office action appears to ignore other recitations from the claims, namely those directed to the ability of both the system handler application and the operating system kernel to link and load gaming program shared objects and device handlers. Claim 71 for example recites "the system handler application then loading a first shared object and providing Application Program Interface functions called by the first shared object" and "the operating system kernel then loading at least one additional device handler and repeating steps (b) through (e) for said at least one additional device handler". The office action makes no mention of where it is believed that such subject matter is taught by Mathur et al.

The kernel (214) in Mathur et al. is distinctly different from the core system interface (220) and the device manager 210, and there is no suggestion that both a system handler application and the operating system kernel can load and execute shared objects and device handlers. In fact, Mathur et al. appears to specifically and exclusively rely on the core system interface 220 to interface with the applications,

> The core system interface 220 is the module through which applications can access the operating system. The core system

> interface 220 includes functions to transfer API calls to the appropriate operating system server process. Mathur et al. 6:47-51.

The device manager module 210 is used to similarly control device drivers:

> Device Manager module 210 is a module that handles installable device drivers. Mathur et al. 9:49-50.

The office action has pointed to no teaching or suggestion of loading and executing directly from both a system handler application and the operating system kernel, and there would appear to be no such teaching in Mathur et al. or anywhere else. As such, the rejection of claim 71 is improper on its face. The rejection of claim 71 and those of claims 72 and 73 depending therefrom are traversed. Reconsideration and allowance are respectfully requested.

In view of the above amendment, applicant believes the pending application is in condition for allowance.

Dated: April 14, 2006

Respectfully submitted,

By _____

Paul B. Stephens
Registration No.: 47,970
MARSHALL, GERSTEIN & BORUN LLP
233 S. Wacker Drive, Suite 6300
Sears Tower
Chicago, Illinois 60606-6357
(312) 474-6300
Attorney for Applicant